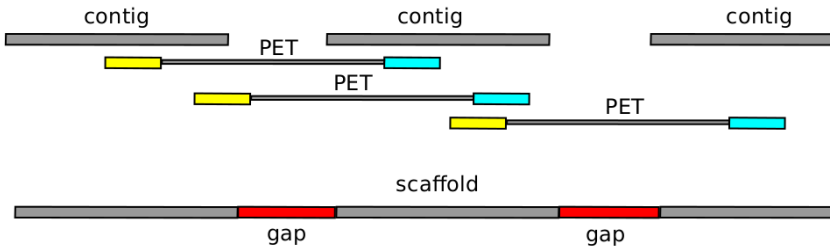
[MBI.A] Asembler DNA, sparowane końce - dokumentacja wstępna

Michał Aniserowicz, Jakub Turek

Opis problemu

Zadanie polega na implementacji aplikacji, która umożliwia tworzenie *scaffoldów* na podstawie dostarczonych zbiorów *contigów* oraz sekwencji PET.



Założenia

W ogólnym przypadku rekonstrukcja sekwencji *contigów* nie jest możliwa. Z tego względu, na potrzeby projektu przyjęto następujące założenia:

- Początki i końce łańcuchów PET to sekwencje unikalne. Wystąpienie takiej sekwencji w jednym z *contigów* oznacza, że jest to odpowiednio początek lub koniec sekwencji PET.
- Badane są wyłącznie takie permutacje *contigów*, dla których wystąpienie początku sekwencji PET implikuje przynajmniej częściowe wystąpienie jej końca w dalszej części łańcucha. Innymi słowy początek lub koniec sekwencji PET nie może w całości wystąpić w przerwie (*gap*) *scaffoldu*.
  - Wyjątkiem od tej reguły jest początek i koniec *scaffoldu*, gdzie mogą występować, odpowiednio, niesparowane końce lub początki sekwencji PET.
- Sekwencje należące do różnych par sparowanych końców mogą częściowo zachodzić na siebie.

Algorytm

Do rozwiązania zadania użyty zostanie algorytm typu brute-force działający według następującego schematu:

1. Wybierana jest początkowa permutacja *contigów*.
2. Dla danej permutacji obliczany jest ranking R:
   - Ranking R określa, dla danej kombinacji *contigów*, liczbę pokrywających się zasad dla zbioru dopasowań sekwencji PET do łańcucha.
3. Sprawdzane jest czy wartość R jest większa niż dotychczas uzyskana maksymalna wartość rankingu. Jeżeli tak, rozwiązanie zachowywane jest jako najlepsze.
4. Algorytm jest powtarzany dla każdej unikalnej permutacji *contigów*.

Sekwencja *contigów* dobierana będzie w sposób losowy. Jako zadanie dodatkowe może zostać przygotowana heurystyczna strategia doboru permutacji.

Implementacja

Projekt zostanie zaimplementowany w języku C#[1] (platforma .NET). Testy jednostkowe zostaną napisane w oparciu o platformę Visual Studio Unit Testing Framework, z użyciem bibliotek Moq[2] oraz AutoMoq[3].

Aplikacja będzie posiadała interfejs okienkowy stworzony w technologii WPF, umożliwiający odczyt danych wejściowych z/zapis danych wyjściowych do pliku. Na dane wejściowe składają się:

- Opis *conitgów* w postaci łańcuchów tekstowych oddzielonych znakami nowej linii.

```
ACAGCTTA
CCGGGTAC
TACAGCTT
```

- Opis sekwencji PET w postaci dwóch sekwencji (początek, koniec) oraz długości łańcucha. Dane w sekwencji oddzielone przecinkami, natomiast kolejne PET'y oddzielone znakami nowej linii.

```
GATC,CCAT,100
GGCT,AGAA,1500
```

Dane wyjściowe to uporządkowana sekwencja *contigów* oddzielonych znakami spacji reprezentującymi długość przerwy (*gap*).

```
CCGGGTAC        TACAGCTT  ACAGCTTA
```

[1] W przypadku, gdy wybór technologii będzie rzutował na obniżenie oceny końcowej (brak przenośności) projekt zostanie zaimplementowany w języku Java.

[2] http://code.google.com/p/moq/

[3] https://github.com/darrencauthon/AutoMoq

Przykład

- Plik wejściowy:

```
ACAGCTTA
CCGGGTAC
TACAGCAA

CCG,TACA,15
GCA,GCT,13
```

- Plik wyjściowy:

```
CCGGGTAC   TACAGCAA   ACAGCTTA
```